SNATI
Seminar Nasional
Aplikasi Teknologi Informasi 2004
Yogyakarta, 19 Juni 2004

Perbandingan Metode-Metode dalam Algoritma Genetika untuk *Travelling Salesman Problem*

Irving Vitra P.
Jurusan Teknik Informatika, Fakultas Teknologi Industri,
Universitas Islam Indonesia, Yogyakarta
e-mail: ipink@engineer.com

Abstrak

The Travelling Salesman Problem, TSP, is a well known and popular problem that has become a standard for testing computational algorithms. The basic problem is that of a salesman working out the minimum distance tour of a number of cities, given their locations. Every city must be visited, but only once and the optimal solution has the lowest total distance.

The aim of this research is to solve the problem with Genetic algorithm and its methods. The result of the research is software whose input is a cartesius coordinate and output is a graph that present the minimum route.

Keywords: *genetic algorithm, chromosom, generation, fitness, crossover, mutation.*

1. Pendahuluan

Pencarian Lintasan Terpendek atau *Travelling Salesman Problem* (TSP), merupakan permasalahan optimasi yang mempunyai banyak terapan penting seperti routing dan penjadwalan produksi. Dalam TSP seseorang harus menentukan suatu bentuk perjalanan, dengan ketentuan orang tersebut berangkat dari suatu kota awal dan mengunjungi semua kota tujuan sekali tanpa mengulangi kembali kota tujuan sebelumnya untuk kembali ke kota awal dengan jarak terpendek.

Untuk menyelesaikan masalah tersebut diatas (pencarian lintasan terpendek) telah dikembangkan beberapa alternatif metode, diantaranya metode *Hill Climbing* (*Simple Hill Climbing* dan *Stepest Ascent Hill Climbing*), *Ant Colony system* dan Metode Pemrograman Dinamis *(Dynamic Programming*).

Dalam pembahasan berikutnya metode yang digunakan ialah Algoritma Genetika (*Genetic Algorithm*). Metode ini digunakan untuk menguji apakah Algoritma Genetika dapat digunakan sebagai alternatif metode dalam optimasi untuk masalah TSP.

1.1 Rumusan Masalah

a. Bagaimana membuat perangkat lunak pada kasus *Travelling Salesman Problem* menggunakan Algoritma Genetika dengan beberapa metode di dalamnya.
b. Bagaimana membuat perangkat lunak yang dapat menerima masukan dengan beberapa variabel yang bersifat fleksibel sehingga hasilnya lebih realistis.

1.2 Tujuan Penelitian

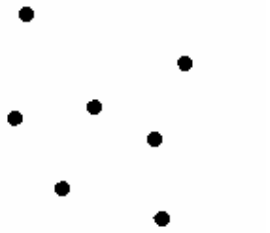
a. Menguji kemampuan Algoritma Genetika beserta metode-metode yang terdapat di dalamnya untuk menyelesaikan TSP.
b. Mengaplikasikan metode-metode itu ke dalam program sederhana untuk mencari lintasan terpendek dari suatu masukan.

2. Dasar Teori

2.1 Travelling Salesman Problem

TSP secara sederhana dapat didefinisikan sebagai proses pencarian lintasan terefisien dan terpendek dari beberapa kota yang dipresentasikan, melewati setiap kota tersebut dan kembali ke kota awal. Setiap kota hanya bisa sekali disinggahi. Persoalan yang dihadapi TSP ialah bagaimana merencanakan total jarak yang minimum. Untuk menyelesaikan persoalan tersebut, tidak mudah dilakukan karena terdapat ruang pencarian dari sekumpulan permutasi sejumlah kota. Maka TSP kemudian dikenal dengan persoalan Non Polinomial.

Gambaran sederhana dari pengertian TSP adalah sebagai berikut:

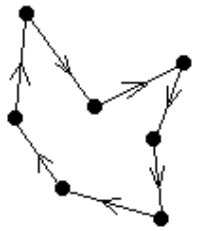


Gambar 1. Posisi kota-kota yang akan dilewati

Kota-kota pada gambar diatas masing-masing mempunyai koordinat (x,y), sehingga jarak antar kedua kota dapat dihitung dengan rumus:

$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Setelah jarak-jarak yang menghubungkan tiap kota diketahui, maka dicari rute terpendek dari jalur yang akan dilewati untuk kembali ke kota awal.



Gambar 2. Rute optimal yang telah dicari

Pencarian Lintasan Terpendek dapat dinyatakan dengan persamaan:

$$Min \sum_{k=1}^{n} \sum_{m=1}^{n} d_{k,m}$$

2.2 Algoritma Genetika

2.2.1 Deskripsi Algoritma Genetika

Algoritma genetika merupakan salah satu algoritma pencarian berstruktur yang didasarkan pada analogi mekanisme seleksi dan informasi genetika alami. Dalam penggunaannya, algoritma genetika meniru beberapa proses yang ditemukan pada evolusi alamiah[1][5].

Dalam algoritma genetika, kromosom-kromosom dalam suatu populasi merupakan set-set solusi yang mungkin dari suatu permasalahan optimasi. Setiap solusi mempunyai fungsi sesuaian (*fitness*) terhadap tujuan yang ingin dicapai. Semakin tinggi nilai sesuaiannya (meminimalkan lintasan) maka akan memiliki peluang yang besar untuk menuju solusi optimum yang dikehendaki (mekanisme roda rolet). Set solusi yang memiliki fungsi *fitness* yang tinggi akan diganti dengan set solusi baru. Set solusi yang baru tersebut dibentuk berdasarkan set solusi sebelumnya yang *fitness*nya maksimal. Dengan demikian algoritma ini melakukan simulasi evolusi pada suatu populasi kromosom. Fungsi evaluasi untuk menghitung nilai *fitness* [3]:

$$fitness = \frac{1}{Dtot}, dengan\ Dtot = panjang\ rute$$

2.2.2 Persilangan (*Crossover*)

CrossOver adalah suatu proses penukaraan gen dari dua individu induk untuk memperoleh dua individu anak. Ini dilakukan dengan harapan agar diperoleh individu-individu anak dengan struktur gen yang lebih baik. Operasi *CrossOver* (perkawinan silang) mengkombinasi ulang gen-gen dari dua kromosom yang diperoleh melalui proses seleksi untuk memperoleh kromosom yang lebih baik.

Pada operasi *CrossOver* dibutuhkan dua induk untuk menghasilkan keturunan (*binary operator*). Untuk itu setiap *CrossOver* diperlukan dua induk dari hasil seleksi, jika hasil seleksi merupakan bilangan ganjil maka dibuang satu kromosomnya sehingga menjadi bilangan genap.

a. Partial Mapped CrossOver(PMX)

Partial Mapped CrossOver (PMX) diperkenalkan oleh Goldberg dan Lingle. PMX menggunakan *procedure* khusus untuk mengatasi terjadinya *offspring* yang tidak legal dari persilangan dua titik. Inti dari PMX adalah persilangan dua titik ditambah dengan prosedur perbaikan[2]. Cara kerja PMX adalah sebagai berikut:

- Pilih dua posisi sepanjang *string* secara acak bebas. *Substring* yang didapat dari dua posisi yang telah dipilih disebut *mapping sections*.
- Tukarkan posisi dari *substring* antara kedua *parent* untuk menghasilkan *protochildren*.
- Analisis *mapping relationship* diantara dua bagian *mapping* tersebut.
- Buat *offsping* menjadi legal jika dari langkah 2 ditemukan *offspring* yang tidak memenuhi kriteria legal (*illegitimate*)

b. Order CrossOver(OX)

Order CrossOver dikenalkan oleh Davis, merupakan variasi dari *PMX* dengan perbedaan pada prosedur pembenahan *CrossOver*. Cara kerja *OX* adalah sebagai berikut :

- Pilih *substring* secara bebas dari *parent*.
- Copy *substring* ke *offspring* yang akan digenerasi dengan posisi yang sama.
- Abaikan gen dengan nilai yang sama dengan yang sudah ada di langkah 2. Hasilkan urutan yang memiliki *substring* yang dibutuhkan oleh *protochild*.

- Tempatkan sisa *substring Parent* lainnya dengan posisi acak ke *protochild* dengan posisi dari kiri ke kanan menurut aturan dalam menggenerasi *offspring*.

c. Position Based CrossOver(PBX)

Position Based CrossOver dikenalkan oleh Syswerda, pada intinya sejenis representasi *CrossOver* pada umumnya dengan prosedur pembenahan. *Position Based* disebut juga variasi dari *OX* dengan pemilihan *substring* secara tidak urut. Cara kerja *Position Based CrossOver* adalah sebagai berikut:

- Pilih satu set posisi *Parent* secara acak.
- Copy *substring* sesuai dengan posisi awalnya ke *protochild* sesuai dengan posisi awal *Parent* asalnya.
- Abaikan *offspring* dengan nilai yang sama dengan yang sudah ada di langkah 2. Hasilkan urutan yang memiliki *substring* yang dibutuhkan oleh *protochild*.
- Tempatkan *substring* dengan posisi acak ke *protochild* dengan posisi dari kiri ke kanan menurut aturan dalam menggenerasi *offspring*.

d. Order Based CrossOver(OBX)

Order Based CrossOver dikenalkan oleh Syswerda, merupakan variasi dari *position based CrossOver*, perbedaanya terletak pada penempatan *substring* dari *Parent* yang pertama yang ditempatkan sesuai order atau *substring* yang belum ada di *protochild* dari posisi yang telah ditempati *substring* dari *Parent* kedua.

e. Cycle CrossOver(CX)

Cycle CrossOver (CX) dikenalkan oleh Oliver, Smith dan Holland. Seperti *position based CrossOver*, CX akan mengambil *substring* dari salah satu *Parent* dan memilih sisanya dari *Parent* lainnya. Perbedaannya adalah *substring* dari *Parent* pertama tidak dipilih secara acak dan urutan node pada *substring* yang dipilih mempunyai hubungan berantai diantara dua *Parent* tersebut. Cara kerja *Cycle CrossOver* adalah sebagai berikut :

- Temukan hubungan antara posisi *substring* diantara kedua *Parent.*
- Copy urutan gen yang sudah terdefinisi (saling terhubung) dalam langkah 1 ke dalam sebuah *child*.
- Tentukan urutan *substring* sisa dari *substring* yang sudah ada di dalam rantai CX.
- Isi posisi *child* dengan sisa *substring*.

2.2.3 Mutasi (*mutation*)

Operasi mutasi melakukan perubahan nilai gen-gen pada kromosom dengan cara menyisipkan kode informasi ke dalam suatu individu. Mutasi bertujuan untuk menjamin bahwa algoritma genetika melakukan ekplorasi pada ruang pencarian sehingga tidak terperangkap ke dalam suatu bagian tertentu. Cara melakukan mutasi tergantung pada jenis pengkodean kromosom[2].

Pada Mutasi ini hanya diperlukan satu induk untuk menghasilkan satu anak (*unary operator*).

a. Inversion Mutation

Inversion Mutation memilih dua posisi gen secara acak dari kromosom secara acak dan membalikkan urutan diantara posisi yang telah terpilih sebelumnya.

b. Insertion Mutation

Insertion Mutation memilih nomor gen secara acak dan melakukan operasi insert secara acak pula ke kromosom tersebut.

c. Displacement Mutation

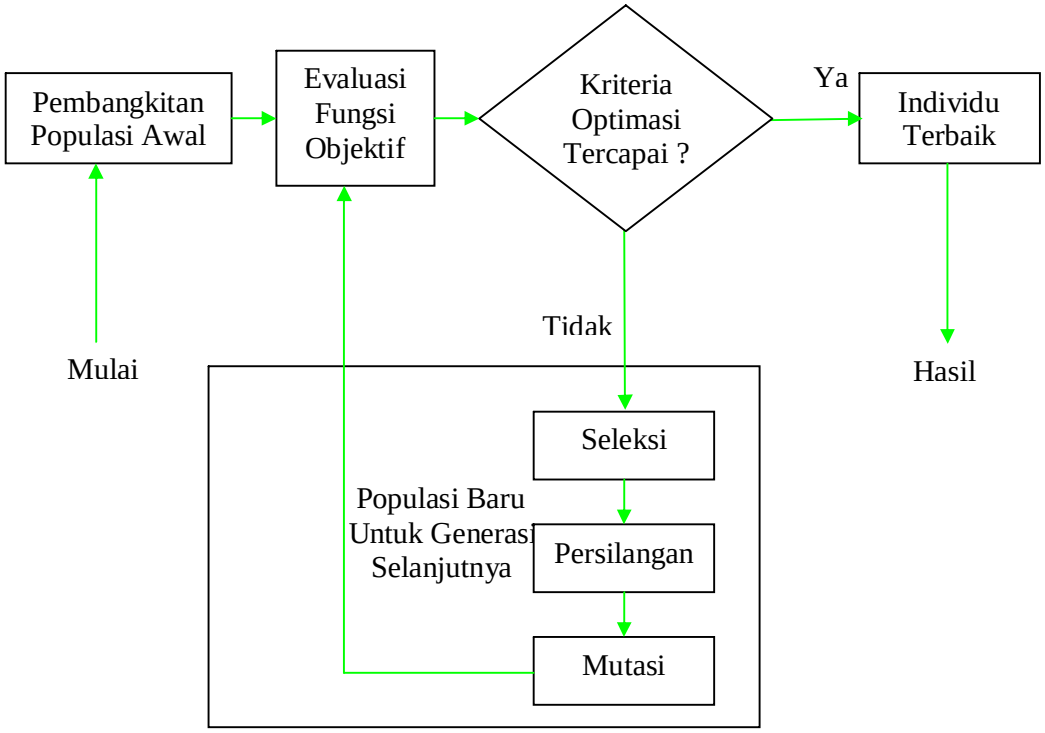
Displacement Mutation akan memilih *substring* secara acak dan melakukan *swapping* (menukar posisi) seperti dalam gambar berkut ini. *Insertion* dapat dikatakan sebagai kondisi tertentu di mana *displacement* hanya mempunyai satu gen.

d. Reciprocal Mutation

Reciprocal Mutation akan memilih dua posisi secara acak dan melakukan *swapping* (menukar posisi).

2.2.4 Pembaharuan Generasi

Proses terakhir setelah operasi seleksi, *CrossOver* (perkawinan silang) dan mutasi (*mutation*) adalah pembaharuan Generasi, yang dilakukan untuk mengganti populasi lama dengan populasi baru hasil seleksi, *CrossOver* dan mutasi, dengan harapan bahwa populasi baru tersebut akan mempunyai nilai *fitness* yang lebih baik daripada populasi lama. Proses pembaharuan generasi yang digunakan adalah *steady state update*, yaitu mempertahankan kromosom yang baik dari generasi lama ke generasi baru dan hanya menggantikan kromosom yang nilai *fitness*nya kurang baik[4].



Gambar 3. Struktur algoritma genetika [4]

3. Pengujian dan Pembahasan

Di dalam pemrosesan data masukan, perangkat lunak ini mampu menangani iterasi sebanyak 10000 kali dengan jumlah populasi maksimum 10000. Sehingga dapat melakukan perhitungan *fitness* dengan variasi kombinasi kromosom sebanyak *n* permutasi dimana *n* merupakan jumlah kota.

Untuk waktu pemrosesan, grafiknya ialah linier (berbanding lurus) untuk setiap jumlah data (panjang kromosom) dan parameter genetik (jumlah populasi dan generasi). Sebelum melakukan optimasi, ditentukan terlebih dahulu model pengkodean sistem (pengkodean gen-gen), dalam hal ini yang digunakan ialah deret (barisan) angka (himpunan angka) sepanjang 1 hingga panjang kromosom. Langkah selanjutnya adalah pembangkitan generasi awal populasi dengan pencuplikan acak gen-gen yang membentuk kromosom. Kromosom-kromosom yang terbentuk selanjutnya dievaluasi dengan penghitungan *fitness* yang didasarkan dengan fungsi jarak.

Di dalam penghitungan ini juga dilakukan penghitungan peluang untuk memperoleh tekanan seleksi yang seragam. Pemilihan induk dilakukan dengan menggunakan roda rolet, di mana semua individu ditempatkan ke dalam suatu lingkaran. Persentase luas dari setiap induk pada lingkaran roda rolet didasarkan nilai *fitness*nya. Semakin besar nilai *fitness* semakin luas pula bagiannya pada roda rolet.

Perbaikan generasi dilakukan dengan mekanisme seleksi, *CrossOver* dan mutasi. Dua induk (atau kelipatannya) dipilih untuk melakukan *CrossOver*. Semetara mutasi dilakukan dengan peloncatan-peloncatan gen-gen induk-induk baru (hasil *CrossOver* induk-induk generasi sebelumnya), tujuannya adalah untuk mempertahankan perbedaan diantara populasi dan mencegah terjadinya konvergensi *premature* (dini).

Setelah itu proses optimasi dimulai dari awal lagi dengan populasi yang baru (hasil operasi algoritma genetika generasi sebelumnya). Proses akan berhenti jika iterasi telah mencapai generasi maksimum.

Tabel 1. Data 19 kota [6]

Kota	***X***	***Y***
1	86	98
2	84	30
3	120	76
4	75	60
5	4	15
6	47	4
7	140	12
8	56	60
9	65	85
10	135	185
11	170	170
12	70	175
13	43	65
14	34	180
15	145	130
16	176	140
17	160	33
18	113	50
19	140	80

Nilai – nilai parameter genetika yang digunakan saat pengujian adalah:

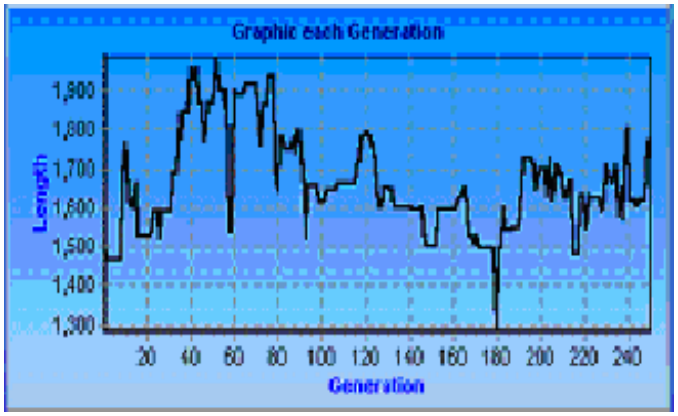
1. Maksimum Generasi = 2000
2. Populasi = 200
3. Probabilitas *CrossOver* = 0.3

4. Probabilitas Mutasi = 0.03
5. Metode *CrossOver* = *Partial Mapped /Order Based / Cycle / Position Based*
6. Metode Mutasi = *Insertion / Reciprocal / Inversion / Displacement*

Data diuji sekali proses tiap kombinasi metode-metodenya. Perbandingan hasil pengujian dapat dilihat di tabel 2.

Tabel 2. Hasil pengujian sistem

Uji Ke	*Metode*	*Rute Hasil Optimasi*	*Panjang Rute*	*Generasi Optimum*	*Waktu Proses*
1	*OX - Inv*	4 13 9 16 11 19 10 12 14 15 3 18 1 17 7 5 6 2 8 4	1195.10	978	52336
2	*OX - Ins*	12 14 5 6 2 4 1 9 13 7 17 15 19 18 8 3 16 11 10 12	**1090.58**	**1164**	**51033**
3	*OX - Dis*	1 19 15 16 3 7 17 18 2 9 6 8 5 13 4 12 14 11 10 1	1194.24	1131	51624
4	*OX - Rec*	6 12 14 10 15 11 16 19 3 17 8 4 2 1 7 18 9 13 5 6	1163.66	1586	51194
5	*OBX - Inv*	15 19 17 7 6 12 14 9 13 4 3 18 2 1 5 8 10 11 16 15	1207.10	1330	53376
6	*OBX - Ins*	12 14 9 3 1 18 4 8 13 2 5 6 7 10 11 16 17 19 15 12	1171.55	1607	53207
7	*OBX - Dis*	17 16 15 12 14 1 10 11 19 3 2 6 4 7 8 5 13 9 18 17	1229.95	396	53737
8	*OBX - Rec*	3 15 11 10 12 14 19 7 17 16 13 4 6 5 9 1 18 2 8 3	1204.70	1885	53837
9	*PBX - Inv*	19 3 4 13 9 18 1 16 15 10 11 12 14 2 6 8 5 7 17 19	1152.02	1009	54088
10	*PBX - Ins*	9 13 6 18 17 19 3 15 16 10 11 14 12 7 5 2 4 8 1 9	1174.77	492	53577
11	*PBX - Dis*	7 11 15 19 16 10 12 14 13 5 6 4 8 9 3 2 1 17 18 7	1199.79	841	53998
12	*PBX - Rec*	3 15 16 11 10 12 1 9 13 6 2 18 5 14 8 4 17 7 19 3	1130.94	1071	53608
13	*CX - Inv*	13 1 11 10 16 15 17 6 5 8 4 2 3 19 18 7 12 14 9 13	1183.84	10	53307
14	*CX - Ins*	5 13 17 3 4 8 6 7 19 16 15 12 14 10 11 18 2 1 9 5	1247.46	12	53627
15	*CX - Dis*	14 12 1 11 10 8 4 5 6 13 9 7 17 18 2 15 3 16 19 14	1346.13	42	54328
16	*CX - Rec*	19 18 4 7 17 2 9 1 12 14 13 3 16 11 10 15 5 6 8 19	1235.54	1893	53677
17	*PMX - Inv*	5 2 15 14 12 1 3 10 11 16 18 4 8 13 9 19 17 7 6 5	1159.24	106	76119
18	*PMX - Ins*	6 7 9 18 4 17 8 2 1 3 19 15 10 16 11 12 14 13 5 6	1220.53	159	68118
19	*PMX - Dis*	8 6 5 1 17 2 13 18 3 19 9 14 12 15 16 11 10 7 4 8	1238.17	974	61859
20	*PMX - Rec*	8 4 9 12 10 16 15 11 14 17 3 1 7 19 18 2 13 6 5 8	1244.97	979	64452



Gambar 4. Salah satu implementasi untuk *OX - Rec*

4. Kesimpulan

Dari hasil pengujian sistem dapat disimpulkan sebagai berikut:

a. Hasil terbaik dari pengujian Data 19 Kota diperoleh pada kombinasi *Order CrossOver* dan *Insertion Mutation* pengujian pertama.
b. Pada pengujian Data 19 Kota, terlihat generasi Optimum tiap satu kombinasi berbeda-beda, ini menunjukkan bahwa hasil optimum belum tentu diperoleh pada akhir generasi.
c. Untuk proses menggunakan metode persilangan PMX, membutuhkan waktu yang lebih lama daripada metode persilangan yang lain. Ini dikarenakan PMX memerlukan langkah-langkah yang lebih banyak sebelum mendapatkan hasil.
d. Algoritma Genetika beserta metode-metode di dalamnya dapat diterapkan pada optimasi masalah TSP sebagai alternatif pencarian solusi dari metode-metode sebelumnya .
e. Hasil dari pemrosesan dengan Algoritma Genetika untuk setiap kali proses belum tentu menghasilkan solusi yang optimum, walaupun dari keseluruhan generasi (untuk 1 kali proses) merupakan solusi yang optimum.
f. Untuk menghasilkan solusi yang benar-benar optimum dibutuhkan proses pencarian yang berulang-ulang karena dengan penggunaan metode persilangan dan mutasi yang lebih banyak (20 kombinasi) dapat memunculkan solusi yang tersembunyi (terpendam).
g. Implementasi Struktur Data Sistem Optimasi ini dibuat dengan Array Dinamis, untuk pengembangan ke depan diharapkan ada Sistem Optimasi dengan Struktur Data List dengan Pointer. Selain itu penentuan fungsi *fitness* yang tepat diharapkan dapat meningkatkan performasi pencarian solusi.
h. *Fitness* terbaik dari Sistem Optimasi ini belum tentu diperoleh pada awal pencarian, untuk pengembangan ke depan diharapkan ada beberapa metode yang digabungkan dengan Sistem ini sebagai Optimasi hasil akhirnya. Sehingga *fitness* terbaik bisa diperoleh pada awal pencarian.

Daftar Pustaka

[1] Beasley, Bull dan Martin, *Genetic Algorithm*, Anonim, Oktober 2000.
[2] Gen, Mitsuo, and Runwei Cheng. *Genetic Algoritms and Engineering*. John Wiley & Sons, Inc, New York, 1997.
[3] Hidayat, Taufik. *Penyelesaian Chromatic Number sebuah Graph dengan Algoritma Genetika*, Yogyakarta, 2001
[4] Indarto, Wawan., *FlowShop Scheduling dengan Algoritma Genetika dengan Pipelining Hybrid Genetic Algorithm*. Tugas Akhir T. Informatika UII, Yogyakarta, 2002
[5] Manongga, D.,Handoko, *Penggunaan Algoritma genetik dalam estimasi Fungsi Non Linear*, Proceeding SITIA 2000, Surabaya, 2000.
[6] Saleh, Chairul., Azmi., Jamaliddin, Yusoff., *Algoritma Genetik untuk Pemecahan Travelling Salesman Problem*, Jurnal *TEKNOIN*, vol.5, No.1, H.33-43, Yogyakarta, 2000.